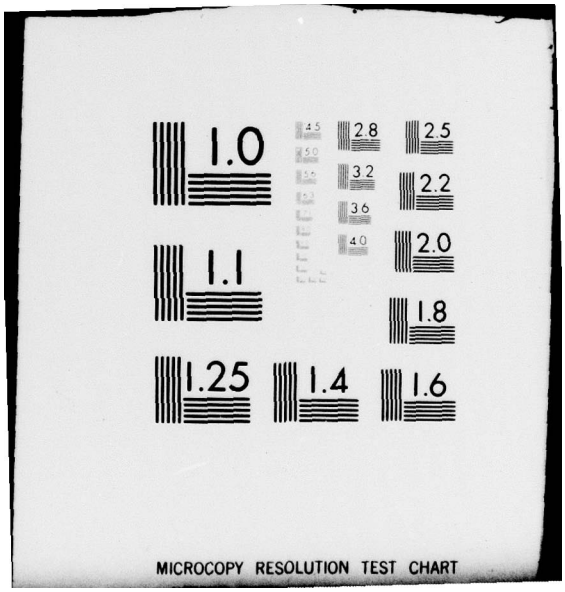
AD-A074 628 RAND CORP SANTA MONICA CA F/G 5/1
USING PERFORMANCE METRICS IN SYSTEM DESIGN,(U)
OCT 78 J LOCKETT
UNCLASSIFIED RAND/P-6242 NL

| OF |
AD
A074628

END
DATE
FILMED
11-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART

(2)

(6) USING PERFORMANCE METRICS IN SYSTEM DESIGN

LEVEL II

(10) JoAnn Lockett

(11) October 1978

(12) 137

(14) RAND P-6242

The Rand Paper Series

Papers are issued by The Rand Corporation as a service to its professional staff. Their purpose is to facilitate the exchange of ideas among those who share the author's research interests; Papers are not reports prepared in fulfillment of Rand's contracts or grants. Views expressed in a Paper are the author's own, and are not necessarily shared by Rand or its research sponsors.

The Rand Corporation
Santa Monica, California 90406

CONTENTS

I. USING PERFORMANCE METRICS IN SYSTEM DESIGN

INTRODUCTION

Complexities of system design are great and often lead designers to be inward looking in their analyses. Knowledge from various fields can be of benefit in designing systems [1]. Management accountants can describe economic effects of delays in closing schedules, psychologists can provide significant insights into the behavioral characteristics of users to complex command syntax, computer performance analysts can provide alternatives to describe and to measure responsiveness of systems. Even in the case of an innovative system design, the designer can employ such approaches to identify incipient problems and create alternatives with increased cost effectiveness. This paper describes how performance metrics can be used effectively to support system design.

Performance metrics are generally employed after systems are implemented. They are used for monitoring and managing the use of systems and for tuning. Seldom are they viewed as being useful in system design. Use of performance metrics on an early prototype system, however, can provide a good deal of information needed to design and produce an effective system. This paper provides an example which describes how system responsiveness of a prototype system was measured and reported and how that information could be used for system design.

This paper appeared in the Proceedings of the Software Quality Assurance Workshop held in San Diego November 15-17, 1978, sponsored by the Association for Computing Machinery.

II. SOME USES OF PERFORMANCE METRICS

DETERMINING USER REQUIREMENTS

A current methodology which can produce well functioning, well designed, and well documented systems on schedule is known as structured design. Structured design entails a great deal of very careful planning. It begins with a statement of requirements and terminates with a detailed specification of code. There are structured walkthroughs and frequent design reviews, with the actual coding left to the very end. Although structured design has been used with success for some military systems [2], it has received only limited use in the commercial world. A major reason it is not used more often is because users do not know or cannot communicate their requirements. Mechanisms, such as performance measurements, which aid in identifying user requirements will allow techniques such as structured design to be more applicable.

Designers encounter many problems in determining user requirements. Not only do users not know what their requirements are, sometimes they can't even guess at them. Alternatively, users may think they know their requirements when, in fact, they do not. A system designer who bases design decisions on perceived but inaccurate requirements can produce a very costly, and possibly ineffective system. This inability to accurately recognize and describe requirements on the part of the users often leads designers to rely on their own best judgements. The user, forced to live with these conditions, must adapt himself to the delivered system. This does not have to be true. We can produce well functioning, responsive systems which satisy user needs.

The user's needs can be measured. He can be informed of trade-offs. He can provide input for decisions in system design. By including the users in discussions during design, the system designer will obtain feedback before he is committed to a particular implementation. By being careful about the order of implementation of the design he can maximize the information he receives by allowing the user to interact with early versions of the system. Providing an early prototype for the user to review can crystallize the user requirements. Instrumenting the prototype can provide hard data on system usage and user behavior.

SYSTEM RESPONSIVENESS

Metrics of system responsiveness can be particularly effective for system design. Users often have difficulty in quantitatively defining responsiveness of a system -- they know only that they want quick response. When forced to communicate such requirements, they often tend to state them intuitively. For example, a common practice is to establish performance criteria for a system in terms of response time such as "response time must be within 2 seconds" or "90% of the responses must be within 3 seconds."

Since commands are usually not of the same complexity, there is no reason to expect the response times be the same. In reality users don't expect them to be. If users of the systems have a sense of the complexity of the commands they issue, they tend to be flexible in their demands of system responsiveness. They are willing to wait, often for very lengthy periods, for what they feel are complicated

commands. Additionally, some users are more concerned with the total amount of time they must spend at a terminal to accomplish a given amount of work than they are with any individual response time.

Experience has shown that the variability of response time effects whether or not the response time is considered satisfactory by users. Variability in response time can affect whether or not a particular response time is satisfactory to users in at least two different ways. First, a user accustomed to a 10-second response for the execution of a particular command often becomes upset when that command, perhaps due to increased load or additional services, requires 30 seconds. However, had the command always used 30 seconds, the user might have found 30 seconds to be a satisfactory time for the response. Second, users become dissatisfied when response time for a particular command varies noticeably within a session. Designers truly sensitive to user satisfaction with the responsiveness of a system will choose metrics which reflect not just specific response times but variations within a session and over time as well.

DESIGN DECISIONS

Many design decisions affect the performance of a system. Often these decisions are made arbitrarily or are based on criteria unrelated to performance. Using performance metrics, such as system responsiveness, in the design stage will provide the designer information on how the parameters (e.g., matrix size, number of allowable cases) affect system performance, allowing him to choose the parameters more intelligently. Alternatively, guidelines can be provided to users to explain what range of parameter values result in very long

response times, leaving the choice to the user about whether elongated response time is justified for a particular interaction.

TRADEOFFS

User satisfaction is not based solely on the functional capability of a system, but on usability, reliability, and performance as well. Often, the user cannot have everything he wants in a system. The final product may be the result of compromise. Certain functional capabilities may be eliminated to achieve specific performance goals or, on the other hand, the user may be willing to sacrifice performance to obtain some functional capability. In either case, it is important to know whether or not both goals are achievable and the costs of achieving each goal.

III. EXAMPLE

The author employed techniques from the field of computer performance analysis on an experimental menu-driven, minicomputer-based information system designed for users to enter, edit, retrieve, manipulate, examine, and analyze clinical patient data [3]. When this work was done the system was a prototype and many decisions had not yet been made about its characteristics and design. (For example, potentially poor response time might be avoided by restrictions on the number of simultaneous users allowed on the system or in certain services being provided only on an overnight basis.) Consequently we needed to define parametric limits on loading and the functions to be provided in the interactive situation; we needed to know system responsiveness.

The prototype system was composed of several activities and subactivities. The user progressed through the system by responding to questions or selecting among alternatives presented on the screen. Accomplishing a useful piece of work could require many inputs by the user. The amount of work to be done by the system in response to user input varied widely. Some responses resulted in little more than the appearance of a new question, others resulted in the execution of analysis packages, reformatting of the data, and presentation of a new display. There were no distinct classes of commands. We needed a metric to show the responsivness of the system and how the responsiveness changes as the system load changes. Reporting the response time for each command would simply confuse, and computing an average would be of even less value. We applied the scripting technique of computer

performance analysis to measure the time required to complete a given amount of work on the system -- the time required to complete a sequence of commands rather than just a single command.

The system could record and store sequences of commands for subsequent execution. We created such command files for each of several commonly used activities. Each command file represented typical use of each activity.

Early versions of the system were in use at Rand and at selected clinical research centers. Feedback from users was actively sought. User inputs were collected at each site and were periodically processed to determine characteristics about system use. We were able to take advantage of these data to select activities to be tested and to build representative scripts.

Each of the remote systems was accessable from Rand via telephone. This was particularly convenient as it allowed us to take measurements as frequently as we liked without letting users know we were doing so. Sometimes just letting users know tests are being conducted changes their behavior and biases system usage [4].

We also employed another technique from computer performance analysis in evaluating online systems -- measuring a defined incremental load on the system. The system itself may be loaded either naturally or artificially,* depending on the objective of the measurement activity.

---

*A natural load on the system is activity on the system caused by real users doing real work. An artificial load is activity on the system which does not accomplish real work. Both have their advantages (and disadvantages). The natural load provides actual system usage. Artificial load provide repeatability and are easier to quantify. Both types of load are used in computer performance analysis. For example, we might test on a naturally loaded system to determine

To determine baseline measurements, we conducted tests on a standalone system (a system unavailable to other users). Each command file was executed several times. Averages and standard deviations of the measurements were computed. Averages of the response times for each activity were used as the base times. Standard deviations were computed to determine that the variability was low. The baseline measurements served two purposes. First, they provided a reference from which to compare measurements made under system loads. Second, the baseline measurements reflected, at least theoretically, the best service the user could expect. If this baseline response time was unsatisfactory to the user, restricting the number of users or services would not be sufficient. Tuning efforts, involving hardware and/or software modifications would be required.

Once baseline measurements were obtained, we addressed the problem of how to report the subsequent measurements -- measurements made after system modifications or measurements made on a loaded system. Reporting the actual time required for the response or the incremental time difference is inadequate; it does not take into account the amount of work accomplished. An additional 15 seconds may be an acceptable degradation in response time for a command requiring 5 minutes to complete but would probably be unsatisfactory for one normally requiring .5 seconds. We chose to report elongation, the per-

---

the effect of adding terminals. On the other hand, an artificial load might be appropriate to test a system modification. Since the artificial load can be fixed, all variations can be attributed to the modification.

cent increase in response time in excess of the baseline measurement. Elongation is computed as

$$100 \times \left\{ \frac{\text{test time}}{\text{base time}} - 1 \right\} .$$

Periodically, and unannounced, we logged onto each prototype system, ran the tests, noting the time and number of active terminals during the test session, and the times required for each of several executions of each command file. Elongation times for each execution were computed. Additional standalone measurements were made each time a new version of the system was released and when modules were modified. These measurements provided instant indication of any performance degradation or improvement which could be attributed to system design. Similar measurements were made to determine the effects of alternative hardware configurations. The measurements were made with the Rand Monitor/Stimulus-Generator (RMS) [5,6], a prototype measurement tool built at Rand. However, similar measurements have been made successfully using a stopwatch [7].

Elongation time indicated to us the type of service a user would have seen if he had chosen to use the system at the time of the test. The measure allowed us easily and quickly to see the relative effect of various loads and to determine any unusual variability among activities with respect to the loading. For example, if most activities exhibited similar percent increases while one was very high, we could make a more intensive study of that one activity to determine the cause of the sensitivity to the load and to ascertain if there

truly was a problem, perhaps in design or coding technique. Using elongation allowed us to track system performance at a high level and expend time and resources only when problems or potential problems surfaced. The information collected was also an indicator of the actual demand on the system and could be used in deciding if more terminals or simultaneous users could be allowed without excessive system degradation.

IV. SUMMARY

Performance metrics can provide valuable information to support effective system design. The example demonstrated how one performance metric (responsiveness) can provide useful data. Measuring responsiveness can

(1) provide information needed to intelligently make tradeoff decisions.

(2) be used to compare alternative hardware configurations and, therefore, can aid in determining the final hardware specifications for a future system.

(3) can be used to evaluate the performance of software modules and to aid in specifying the design of data structures and application software.

(4) can be used to resolve operational considerations. Measuring the performance effects of certain capabilities can provide data on which to base decisions such as whether or not those capabilities should be provided or if they should be provided, say, on an overnight basis.

Other performance metrics can also provide valuable data and should not be ignored. For example, Boise [8] has shown how quantifying system usage can aid in selecting system features to be retained, deleted, or modified. Availability of early prototypes and application of performance metrics to prototypes and evolving systems can be an effective technique for designing and producing high quality software.

REFERENCES

1. Bell, Thomas E., "Graphical Analysis Procedures for Simulation and Scheduling," Unpublished doctoral dissertation, UCLA 1968.

2. Brown, John R., "Impact of MPP on System Development," TRW, RADC-TR77-121, May 1977.

3. Groner, G. F., M. D. Hopwood, N. A. Palley, W. L. Sibley, "An Introduction to the CLINFO Data Management and Analysis System," The Rand Corporation, R-1541-NIH, December 1977.

4. Shetler, A. C., "Human Factors in Computer Performance Analysis," The Rand Corporation, P-5128, 1974.

5. Bell, Thomas E., and JoAnn Lockett, "A Simple Technique for Controlled On-line System Stimulation," AFIPS Conference Proceedings, Vol. 44, 1975 National Computer Conference, pp. 831-837.

6. Yoshimura, R., "The Rand Monitor Stimulus-Generator: Hardware Implementation," The Rand Corporation, R-1714-PR, November 1975.

7. Lockett, JoAnn, "Computer Performance Analysis in Mixed On-line/ Batch Workloads," AFIPS Conference Proceeding, Vol. 43, 1974 National Computer Conference, pp. 671-676.

8. Boies, S. J., "User Behavior on an Interactive Computer System," IBM Systems Journal, 13, No. 1, pp. 2-18 (1974).